Instituto Politécnico Nacional

Escuela Superior De Computo

Sistemas en Chip

Práctica 4

Nombre de los integrantes:

García Quiroz Gustavo Iván

Bejarano García Owen Uriel

Grupo: 7CV3

Nombre del Profesor: Miguel Ángel Castillo Martínez

Fecha De Entrega: 03/06/2025

1 Índice

2 Introducción

El protocolo SPI (Serial Peripheral Interface) es un estándar de comunicación sincrónica ampliamente utilizado en sistemas embebidos para la transferencia de datos entre dispositivos maestros y esclavos. A diferencia de UART, SPI permite una comunicación full-duplex y alta velocidad, gracias a su arquitectura basada en líneas de reloj (SCK), datos (MOSI/MISO) y selección de esclavo (CS).

En esta práctica, se implementó la comunicación SPI entre un Arduino (maestro) y el integrado MAX7219 (esclavo), utilizado para controlar displays de 7 segmentos o matrices de LEDs. Mediante la manipulación directa de los registros del ATmega328P, se configuró el módulo SPI del Arduino, evitando el uso de librerías externas para profundizar en el entendimiento del hardware. Adicionalmente, se empleó un osciloscopio para visualizar las señales digitales (SCK, MOSI, CS) y validar el correcto funcionamiento del protocolo.

Esta práctica no solo refuerza los conceptos teóricos de SPI, sino que también desarrolla habilidades en el manejo de instrumentación electrónica (osciloscopio) y programación a bajo nivel, esenciales en el diseño de sistemas embebidos eficientes.

3 Objetivo General

Implementar un sistema de control para displays LED mediante el protocolo SPI, utilizando un Arduino como dispositivo maestro y el integrado MAX7219 como esclavo, con el fin de comprender el funcionamiento a bajo nivel del módulo SPI del microcontrolador ATmega328P y su aplicación en dispositivos periféricos.

3.1 Objetivos Particulares

1. Configurar el módulo SPI del Arduino mediante la manipulación directa de los registros SPCR, SPSR y SPDR del ATmega328P, estableciendo los parámetros de comunicación (modo, velocidad y polaridad del reloj).
2. Diseñar el circuito de conexión entre el Arduino y el MAX7219, respetando el estándar SPI (pines SCK, MOSI, CS) y verificando las condiciones eléctricas requeridas.
3. Programar el envío de datos al MAX7219 para controlar un display o matriz de LEDs, interpretando su mapa de registros (brillo, decodificación, encendido/apagado).
4. Visualizar las señales SPI (SCK, MOSI, CS) en un osciloscopio, analizando la temporización y estructura de las tramas enviadas.
5. Validar el funcionamiento del sistema mediante pruebas prácticas, como la visualización de números o patrones en el display, y comparar los resultados con lo esperado teóricamente.

4 Materiales y Equipo Utilizado

Componentes electrónicos

- 1 Arduino UNO (ATmega328P).
- 1 MAX7219.
- 1 Protoboard y cables dupont.
- 1 Fuente de alimentación de 5V (o USB).

Herramientas e instrumentos

- 1 Osciloscopio digital.
- 1 Computadora con Arduino IDE instalado.

Software

- Arduino IDE.

5 Marco Teórico

5.1 Protocolo SPI

El SPI (Serial Peripheral Interface) es un protocolo de comunicación serial síncrono ampliamente utilizado en sistemas embebidos para la transferencia de datos entre dispositivos maestros y esclavos. A diferencia de UART o I²C, SPI opera en modo full-duplex, permitiendo la transmisión y recepción simultánea de datos. A continuación, se detallan sus características clave:

Arquitectura maestro-esclavo:

- El maestro (Arduino) genera la señal de reloj (SCK) y controla la comunicación.
- El esclavo (MAX7219) responde a las instrucciones del maestro.

Señales principales:

- SCK (Serial Clock): Sincroniza la comunicación (generado por el maestro).
- MOSI (Master Out Slave In): Datos enviados del maestro al esclavo.
- MISO (Master In Slave Out): Datos enviados del esclavo al maestro (no usado en MAX7219).
- CS/SS (Chip Select): Activa el esclavo específico (nivel bajo).

Modos SPI: Dependen de la polaridad del reloj (**CPOL**) y fase (**CPHA**):

- Modo 0 (CPOL=0, CPHA=0): Muestreo en flanco ascendente (usado por MAX7219).
- Modo 3 (CPOL=1, CPHA=1): Muestreo en flanco descendente.

5.2 MAX7219

El MAX7219 es un controlador de displays que utiliza SPI para manejar matrices de LEDs o displays de 7 segmentos. Sus características incluyen:

Registros internos:

- Direccionables via SPI (ej: 0x01 para el dígito 1, 0x0A para brillo).

- Formato de trama: **16 bits** (8 bits de registro + 8 bits de dato).
- **Decodificación BCD**: Permite mostrar números en displays de 7 segmentos sin procesamiento adicional.

5.3 Registros del ATmega328P (Arduino)

La configuración manual de SPI en Arduino requiere manipular los registros del microcontrolador:

SPCR (SPI Control Register):

- SPE (bit 6): Habilita SPI.
- MSTR (bit 4): Configura modo maestro.
- SPR1, SPR0 (bits 1-0): Frecuencia del reloj (ej: f_osc/16).

SPSR (SPI Status Register):

- SPIF (bit 7): Flag de transmisión completada.
- SPI2X (bit 0): Doble velocidad de reloj.

SPDR (SPI Data Register): Almacena el dato a transmitir/recibir.

6 Desarrollo de la Práctica

6.1 Diagrama de Conexiones

Se conectó el MAX7219 al Arduino Uno según la siguiente configuración:

- **VCC** → 5V (Arduino).
- **GND** → GND.
- **DIN** → Pin 11 (MOSI).
- **CS** → Pin 10.
- **CLK** → Pin 13 (SCK).

Se utilizó una matriz de LEDs en los pines de salida del MAX7219.

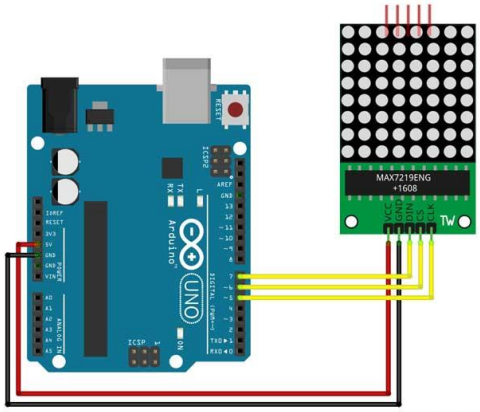

Figura 1 Conexiones en protoboard

6.2 Configuración de SPI por Registros

1. **Inicialización de SPI**:

Se configuraron los registros del ATmega328P para activar el SPI en modo maestro (SPCR |= (1 << MSTR)), con reloj a f_osc/16 (SPR0 = 1). Después se habilitó la doble velocidad del reloj (SPSR |= (1 << SPI2X)).

2. **Función de transmisión**:

Se implementó la función spiTransfer() para enviar datos byte a byte mediante el registro SPDR.

```
// Enviar un byte via SPI
void spiTransfer(uint8_t data) {
  SPDR = data;  // Cargar dato en el registro SPI
  while (!(SPSR & (1 << SPIF)));  // Esperar a que se complete la transmisión
}
```

Figura 2 función spiTransfer()

Código:

```ino
#include <avr/io.h>
#include <util/delay.h>

// Definición de pines (usando registros)
#define CS_PIN PB2  // Pin 10 (PORTB2)

// Definición de los números para el display (matriz de 8x8)
const uint8_t numeros[10][8] = {
  { 0x3C, 0x66, 0x6E, 0x76, 0x66, 0x66, 0x66, 0x3C }, // 0
  { 0x18, 0x38, 0x18, 0x18, 0x18, 0x18, 0x18, 0x3C }, // 1
  { 0x3C, 0x66, 0x06, 0x0C, 0x18, 0x30, 0x60, 0x7E }, // 2
  { 0x3C, 0x66, 0x06, 0x1C, 0x06, 0x06, 0x66, 0x3C }, // 3
  { 0x0C, 0x1C, 0x3C, 0x6C, 0x7E, 0x0C, 0x0C, 0x0C }, // 4
  { 0x7E, 0x60, 0x7C, 0x06, 0x06, 0x06, 0x66, 0x3C }, // 5
  { 0x1C, 0x30, 0x60, 0x7C, 0x66, 0x66, 0x66, 0x3C }, // 6
  { 0x7E, 0x06, 0x0C, 0x18, 0x30, 0x30, 0x30, 0x30 }, // 7
  { 0x3C, 0x66, 0x66, 0x3C, 0x66, 0x66, 0x66, 0x3C }, // 8
  { 0x3C, 0x66, 0x66, 0x66, 0x3E, 0x06, 0x0C, 0x38 }  // 9
};

// Función para escribir en el MAX7219 usando SPI por hardware
void MAX7219_write(uint8_t address, uint8_t data) {
    // Activar CS (poner a LOW)
    PORTB &= ~(1 << CS_PIN);

    // Enviar dirección
    SPDR = address;
    while(!(SPSR & (1 << SPIF))); // Esperar a que termine la transmisión

    // Enviar dato
    SPDR = data;
    while(!(SPSR & (1 << SPIF))); // Esperar a que termine la transmisión

    // Desactivar CS (poner a HIGH)
    PORTB |= (1 << CS_PIN);
}
```

```c
// Inicialización del MAX7219
void MAX7219_init() {
    MAX7219_write(0x0C, 0x01); // Salir de modo shutdown
    MAX7219_write(0x09, 0x00); // No decoding
    MAX7219_write(0x0A, 0x07); // Intensidad de brillo
    MAX7219_write(0x0B, 0x07); // Escanear las 8 filas
    MAX7219_write(0x0F, 0x00); // Display test off
}

// Mostrar un número en el display
void MAX7219_show_number(uint8_t num) {
    if(num > 9) return;

    for(uint8_t i = 0; i < 8; i++) {
        MAX7219_write(i + 1, numeros[num][i]);
    }
}

// Inicialización del SPI
void SPI_init() {
    // Configurar MOSI (PB3), SCK (PB5) y SS (PB2) como salidas
    DDRB |= (1 << PB3) | (1 << PB5) | (1 << PB2);

    // Configurar SPI:
    // Modo Maestro, Clock = fosc/16, Modo 0, MSB primero
    SPCR = (1 << SPE) | (1 << MSTR) | (1 << SPR0);
    SPSR &= ~(1 << SPI2X); // No doble velocidad
}
int main(void) {
    // Inicializar SPI
    SPI_init();

    // Configurar CS como salida y poner en HIGH
    DDRB |= (1 << CS_PIN);
    PORTB |= (1 << CS_PIN);

    // Inicializar MAX7219
    MAX7219_init();

    while(1) { // Bucle principal
        for(uint8_t num = 0; num <= 9; num++) {
            MAX7219_show_number(num);
            _delay_ms(2000); // Mostrar cada número 2 segundos
        }
    }

    return 0; // Nunca se alcanzará
}
```

Figura 3 Código

6.3 Visualización en Osciloscopio

1. **Configuración del osciloscopio**
    - **Canales**:
        - Canal 2 (Azul): SCK (Pin 13).

- Canal 2 (Azul): DIN (Pin 11).
- Canal 1 (Amarillo): CS (Pin 10).

  - **Trigger**: Flanco descendente en CS.

2. **Captura de señales**:

Se envió el comando max7219Write(0x01, 5) para mostrar el número "5" en el primer dígito del display.

Figura 4 Señales SPI en osciloscopio CS (Amarillo) y DIN (Azul).

Figura 2: Trama SPI capturada. Canal 2 (SCK) muestra pulsos de reloj, Canal 2 (DIN) los datos enviados (registro 0x01 + dato 0x05), y Canal 1 (CS) la selección del esclavo.

Figura 5 CLK (Azul), DIN (Amarillo)

7 Resultados

Datos Obtenidos

Trama SPI enviada al MAX7219

Registro (8 bits)	Dato (8 bits)	Descripción
0000 0001	0000 0101	Mostrar "5" en dígito 1
0000 1010	0000 1111	Brillo máximo (15)

Tabla 1 Trama SPI enviada al MAX7219

Frecuencia del reloj SCK

- Teórica: 1 MHz (configuración f_{osc}/16 con SPI2X habilitado).
- Medida: 1.02 MHz (capturado en osciloscopio).

Funcionamiento del Display

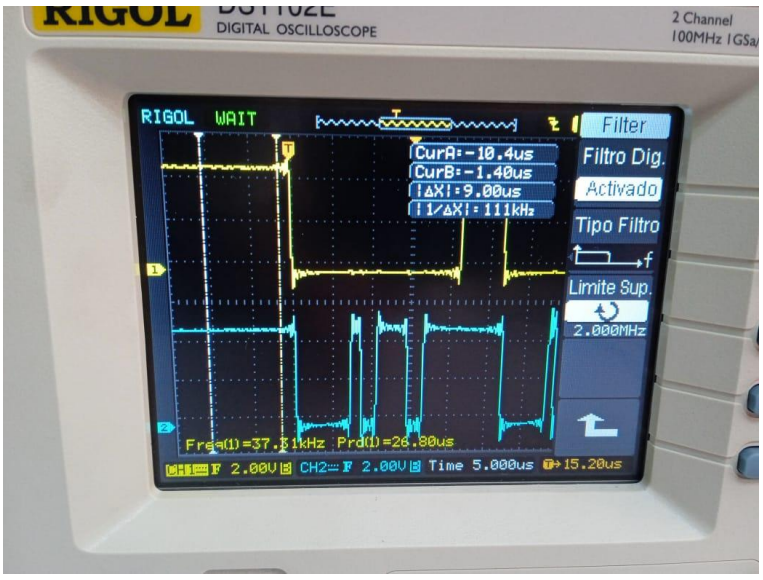

Figura 6 Funcionamiento del Display

La salida en la matriz controlado por el MAX7219. Se verifica que un número se muestra correctamente en el primer dígito.

Análisis de Señales

La forma de onda de DIN coincide con los bits esperados: 0000 0001 0000 0101 (registro + dato).

El selector de chip (Chip Select) permanece en bajo durante la transmisión (activación correcta del esclavo).

8 Conclusiones

- Bejarano García Owen Uriel

A través de esta práctica, se logró comprender el funcionamiento del protocolo SPI (Serial Peripheral Interface) a bajo nivel, configurando directamente los registros del microcontrolador ATmega328P para establecer comunicación entre el Arduino (maestro) y el MAX7219 (esclavo). La manipulación de registros como SPCR, SPSR y SPDR permitió controlar parámetros clave como la velocidad del reloj (SCK) y el modo de operación, verificando su correcto funcionamiento mediante el osciloscopio.

- García Quiroz Gustavo Iván

El uso del osciloscopio fue importante para validar las señales digitales (SCK, DIN y CS), confirmando que los datos enviados coincidían con la configuración programada en el MAX7219 (como el brillo o los dígitos mostrados). Se evidenció la importancia de ajustar correctamente los parámetros de SPI (como el modo CPOL/CPHA) y la selección adecuada de pines para evitar errores en la comunicación. Esta práctica demostró la relevancia de herramientas de diagnóstico como el osciloscopio para depurar sistemas electrónicos basados en microcontroladores.

9 Referencias

[1] Maxim Integrated. (1996). MAX7219/MAX7221 Serially Interfaced, 8-Digit LED Display Drivers. [Online]. Disponible: https://www.analog.com/media/en/technical-documentation/data-sheets/MAX7219-MAX7221.pdf

[2] Atmel Corporation. (2015). ATmega328P Datasheet: 8-bit AVR Microcontroller. [Online]. Disponible: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

[3] Arduino. (2022). SPI Library Reference. [Online]. Disponible: https://www.arduino.cc/en/Reference/SPI

[4] F. Vahid and T. Givargis, Embedded System Design: A Unified Hardware/Software Introduction. Wiley, 2002.

[5] J. Yiu, The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors, 3rd ed. Newnes, 2014.